ELSEVIER

Theoretical Computer Science 276 (2002) 245–259

Theoretical Computer Science

www.elsevier.com/locate/tcs

Decidability of EDT0L structural equivalence ☆

Kai Salomaa[a], Sheng Yu[b]

[a] *Department of Computing and Information Science, Queen's University, Kingston, Ontario Canada K7L 3N6*

[b] *Department of Computer Science, University of Western Ontario, London, Ontario Canada N6A 5B7*

Received January 1997; received in revised form February 2000; accepted March 2000

Communicated by B. Rovan

Abstract

We show that a tree pushdown automaton can verify, for an arbitrary nondeterministically constructed structure tree t, that t does not correspond to *any* valid derivation of a given EDT0L grammar. In this way we reduce the structural equivalence problem for EDT0L grammars to deciding emptiness of the tree language recognized by a tree pushdown automaton, i.e., to the emptiness problem for context-free tree languages. Thus we establish that structural equivalence for EDT0L grammars can be decided effectively. The result contrasts the known undecidability result for ET0L structural equivalence. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Formal languages; Parallel grammars; Tree pushdown automata

1. Introduction

Context-free type grammars G_1 and G_2 are said to be structurally equivalent if, corresponding to each syntax tree of G_1 producing a terminal word, the grammar G_2 has a syntax tree with the same structure, and vice versa. The structure of a syntax tree t is the leaf-labeled tree obtained from t by removing the nonterminals labeling the internal nodes. The importance of the notion of structural equivalence for context-free grammars is due to the fact that it can be decided effectively [13, 15, 19] whereas language equivalence is undecidable.

☆ This work has been supported by the Natural Sciences and Engineering Research Council of Canada grants OGP0041630, OGP0147224, and the Academy of Finland grant 14018. A preliminary version of this paper appeared in the Proceedings of the First Conference of the Centre for Discrete Mathematics and Theoretical Computer Science, DMTCS'96, Auckland, New Zealand, 9–13 December 1996.

E-mail addresses: ksalomaa@cs.queensu.ca (K. Salomaa), syu@csd.uwo.ca (S. Yu).

PII: S0304-3975(01)00197-9

Structural equivalence remains decidable also for parallel context-free (E0L) grammars [16–18, 26]. Surprisingly it was shown in [24] that when the parallel derivations are controlled by a finite set of tables (in an ET0L grammar), structural equivalence is undecidable. We cannot even decide whether an E0L grammar and an ET0L grammar having just two tables are structurally equivalent.

Here we show that structural equivalence becomes decidable if the tables of the grammars are restricted to be homomorphisms, that is, we have EDT0L grammars. Thus for the structural equivalence problem we cross the borderline between undecidable and decidable when restricting the tables of the grammar to be homomorphisms. Our proof uses automata theoretic methods but differs considerably from the automata theoretic decidability proofs for E0L structural equivalence [26] and ET0L strong structural equivalence [11]. The E0L structure trees, as well as ET0L structure trees augmented with the information about the control-sequence used, can be recognized deterministically bottom-up using a tree automaton model for which equivalence is decidable. This appears not possible for EDT0L structure trees since the arbitrary choice of the sequence of tables makes the derivation inherently nondeterministic.

We reduce EDT0L structural equivalence to the emptiness problem for the tree pushdown automata of Guessarian [9]. These automata recognize exactly the context-free tree languages and emptiness can be decided algorithmically. We show that, given EDT0L grammars G_1 and G_2, a tree pushdown automaton can verify that a nondeterministically guessed structure tree of G_1 does not correspond to any valid derivation of G_2.

The decidability proof relies strongly on nondeterminism and an actual algorithm following the proof requires multiple exponential time. It is seen easily that EDT0L structural equivalence is PSPACE-hard, so one cannot expect to find a very efficient algorithm. It has been shown in [25] that E0L structural equivalence is hard for deterministic exponential time. For the EDT0L case we have not obtained an exponential time lower bound.

2. Preliminaries

We assume that the reader is familiar with the basics of formal language theory [30]. We briefly recall some definitions concerning parallel context-free type grammars and tree automata. For more information regarding parallel grammars the interested reader is asked to consult [22], and regarding tree automata we refer the reader to [6, 7].

The cardinality of a finite set A is denoted by $\#A$ and the power set of A is $\wp(A)$. Sometimes we identify a singleton set $\{a\}$ with a. The sets of positive and nonnegative integers are denoted, respectively, by $\mathbb{N}$ and $\mathbb{N}_0$.

The set of (nonempty) finite words over A is A^* (A^+) and λ denotes the empty word. The length of $w \in A^*$ is $|w|$. Let A_i, $i=1,2$, be finite sets. We define the mappings $\Pi_i^{A_1 \times A_2} : (A_1 \times A_2)^* \to A_i^*$, $i=1,2$, by setting for $w=(a_1,b_1)\ldots(a_m,b_m)$,

$m \geqslant 0, a_1, \ldots, a_m \in A_1,\ b_1, \ldots, b_m \in A_2,$

$$\Pi_1^{A_1 \times A_2}(w) = a_1 \ldots a_m, \quad \Pi_2^{A_1 \times A_2}(w) = b_1 \ldots b_m.$$

A *tree domain* D [7] is a nonempty finite subset of $\mathbb{N}^*$ that satisfies the following two conditions:

(i) If $u \in D$, then every prefix of u is in D.

(ii) For every $u \in D$ there exists $\mathrm{rank}_D(u) \in \mathbb{N}_0$ such that $ui \in D$ for $i = 1, \ldots, \mathrm{rank}_D(u)$ and $ui \notin D$ for $i > \mathrm{rank}_D(u)$. (If $\mathrm{rank}_D(u) = 0$, the node u has no successors.)

An *A-labeled tree* is a mapping $t : \mathrm{dom}(t) \to A$, where $\mathrm{dom}(t)$ is a tree domain. A node $u \in \mathrm{dom}(t)$ is said to be labeled by $t(u) \in A$. A node v is a *successor* (respectively, an immediate successor) of $u \in \mathrm{dom}(t)$ if $v = ux,\ x \in \mathbb{N}^*$ (respectively, $x \in \mathbb{N}$). We assume that notions such as the height, the root, a leaf, an internal node, and a subtree of a tree t are known. The height of t is denoted $\mathrm{hg}(t)$. We use the convention that the height of a one-node tree is zero. The subtree of t at node u is t/u.

By the *level* of a node $u \in \mathrm{dom}(t)$ we mean the distance of u from the root, i.e., $|u|$. Clearly, the maximal level of a node of t is $\mathrm{hg}(t)$. The tree t is said to be *balanced* if all leaf nodes of t have the same level. The set of level k subtrees of t, $\mathrm{sub}_k(t)$, $0 \leqslant k \leqslant \mathrm{hg}(t)$, is defined as

$$\mathrm{sub}_k(t) = \{t/u \mid u \in \mathrm{dom}(t),\ |u| = k\}.$$

Note that $\mathrm{sub}_k(t)$ is a set of trees as opposed to a set of occurrences of subtrees. Thus $\mathrm{sub}_k(t)$ contains only one copy of each tree t' that occurs as a subtree defined by a level k node of t. Also $\mathrm{sub}_0(t) = \{t\}$.

In our decidability proof we use the *top-down tree pushdown automaton* model of Guessarian [9]. Below we give a brief informal description of this model which will be sufficient for our purposes. An interested reader can find the formal algebraic definition in [7, 9]. A tree pushdown automaton **A** is an extension of a finite tree automaton where each copy of the finite-state control has access to an auxiliary pushdown store. The automaton begins the computation at the root of the input tree, in a given initial state q_0 and with an initial symbol in the pushdown store. When being in a state q at a node u labeled by b in the input tree and having Z as the topmost stack symbol, depending on the tuple (q, b, Z), **A** can either (i) change the internal state and pop Z from the stack, (ii) change the state and push some symbol to the stack, or (iii) go to the m immediate successor nodes of u in some states $q_1, \ldots, q_m$ sending to each of the m nodes a copy of the pushdown stack. (The node u is assumed to have rank m.) The automaton accepts an input tree t if (in some nondeterministic computation) it reaches all leaves of t in an accepting state. The tree language recognized by **A** is denoted $T(\mathbf{A})$.

In general, the automata of [9] can employ a tree structure in the stack. The automaton model described above is the so-called restricted tree pushdown automaton of [9]. Both the restricted and general tree pushdown automata recognize exactly the

family of context-free tree languages [3–5, 7, 21]. Given a context-free tree language T (in terms of a tree pushdown automaton or a context-free tree grammar), we can effectively construct an indexed grammar that generates the yield of T. Since emptiness is decidable for indexed grammars [1, 2], we have the following result.

Proposition 2.1. *We can decide effectively whether the tree language recognized by a given tree pushdown automaton is empty.*

3. Structural equivalence

We recall and invent some notations concerning parallel context-free grammars [22] and the structure trees of their derivations [24, 25].

An *ET0L grammar* G is a tuple

$$G = (V, \Sigma, s_0, H), \tag{1}$$

where V is a finite alphabet of nonterminals, Σ is a finite alphabet of terminals, $s_0 \in V$ is the initial nonterminal, and H is a finite set of *tables.* A table $h \in H$ is a finite set of rewrite rules $a \to w$, where $a \in V$ and $w \in (V \cup \Sigma)^*$. (We do not allow rewriting of terminals.) The grammar G is an EDT0L (deterministic ET0L) grammar if every table $h \in H$ contains exactly one rule with left side a, for each nonterminal $a \in V$. Thus, h is a morphism $V^* \to (V \cup \Sigma)^*$ and $h(a)$ denotes the right side of the rule of h having the nonterminal a as the left side. The grammar G is an E0L grammar if H contains only one table. The grammar is said to be *propagating* if the right side of any production is not the empty word, i.e., for all $h \in H$ and $a \in V$: $(a \to \lambda) \notin h$.

In this paper we will be dealing mainly with the structure trees of EDT0L derivations. Below we define structure trees for arbitrary ET0L grammars since the definition is not essentially simpler in the deterministic case. In the remainder of this section, G is always an ET0L grammar as in (1).

Let F_G denote the set of $(V \cup \Sigma \cup \hat{\lambda})$-trees. Here $\hat{\lambda}$ is a new symbol that will be used to label nodes corresponding to the empty word. We define the parallel derivation relation of G, $\to_G^{\text{par}} \subseteq F_G \times F_G$, as the union of relations $\to_{G,h}^{\text{par}} \subseteq F_G \times F_G$, $h \in H$, defined as follows. Let $t_1, t_2 \in F_G$ and let $h \in H$. Then $t_1 \to_{G,h}^{\text{par}} t_2$ if and only if t_2 is obtained from t_1 as follows. Assume that t_1 has m leaves $u_1, \ldots, u_m$, where $t_1(u_i) \in V \cup \{\hat{\lambda}\}$, $i = 1, \ldots, m$. (Note that if some leaf of t_1 is labeled by a terminal, the derivation cannot be continued from t_1.) For each $i = 1, \ldots, m$ such that $t_1(u_i) \neq \hat{\lambda}$ choose a rule

$$t_1(u_i) \to a_1^i \ldots a_{k_i}^i \in h, \tag{2}$$

$a_j^i \in V \cup \Sigma$, $j = 1, \ldots, k_i$, $k_i \geqslant 0$. If $t_1(u_i) \neq \hat{\lambda}$ and $k_i \geqslant 1$, then in t_2 the node u_i has k_i successors labeled, respectively by the symbols $a_1^i, \ldots, a_{k_i}^i$. If $t_1(u_i) \neq \hat{\lambda}$ and $k_i = 0$, then in t_2 the node u_i has one successor labeled by the symbol $\hat{\lambda}$. If $t_1(u_i) = \hat{\lambda}$, then u_i is a leaf of t_2.

We denote by t_0 the singleton tree with the only node labeled by the initial nonterminal s_0. The set of *syntax trees* $S(G)$ of G is defined by

$$S(G) = \{t \in F_G \mid t_0 \ (\rightarrow_G^{\text{par}})^* \ t\}.$$

A syntax tree $t \in S(G)$ is *terminal* if all leaves of t are labeled by elements of $\Sigma \cup \hat{\lambda}$. The set of terminal syntax trees of G is denoted $TS(G)$.

In case G is an EDT0L grammar, the rule (2) is determined uniquely by the table h. We call words over the alphabet H *control-sequences*. Given an EDT0L grammar G and a control-sequence $\omega = h_1 \dots h_n$, $n \geqslant 1$, $h_i \in H$, $i = 1, \dots, n$, we denote by $G(\omega)$ the syntax tree obtained from the initial nonterminal by applying the sequence of tables specified by ω. Thus $G(\omega)$ is the unique syntax tree t such that

$$t_0 \ (\rightarrow_{G,h_1}^{\text{par}} \circ \cdots \circ \rightarrow_{G,h_n}^{\text{par}}) \ t, \tag{3}$$

if a tree t as in (3) exists, and otherwise $G(\omega)$ is undefined.

If G is propagating, then in every syntax tree t of G all paths from the root to a leaf have the same length, i.e., t is balanced. For non-propagating grammars all paths from the root to a leaf labeled by an element of $V \cup \Sigma$ have the same length. Note that our definition does not allow the rewriting of terminal symbols, i.e., we assume that the grammars are synchronized [22]. This is not a restriction since an arbitrary E(D)T0L grammar can be easily transformed into a synchronized E(D)T0L grammar in such a way that the transformation preserves structural equivalence of grammars. The transformation (for E0L grammars) is explained in [16].

For $t \in F_G$ denote by w_t ($\in (V \cup \Sigma \cup \{\hat{\lambda}\})^+$) the word obtained by catenating, in the natural left-to-right order of the leaves, the labels of the leaves of t. The *yield* of a syntax tree t is defined as

$$\text{yield}(t) = e_\lambda(w_t),$$

where $e_\lambda : (V \cup \Sigma \cup \{\hat{\lambda}\})^* \to (V \cup \Sigma)^*$ is the morphism defined by setting $e_\lambda(a) = a$ for $a \in V \cup \Sigma$, and $e_\lambda(\hat{\lambda}) = \lambda$.

For a syntax tree t, $\text{yield}(t)$ is the sentential form generated by G in the derivation corresponding to t. The language $L(G)$ generated by G consists of the terminal words generated by G, i.e.,

$$L(G) = \{\text{yield}(t) \mid t \in TS(G)\}.$$

Clearly the above definition is equivalent to the standard definition of the language generated by an E(D)T0L grammar [22].

The *structure of a syntax tree* $t \in S(G)$, $\mathbf{str}(t)$ is the tree obtained from t by relabeling each internal node with σ, where σ is a new symbol not in $V \cup \Sigma$. We denote

$$STS(G) = \{\mathbf{STR}(t) \mid t \in TS(G)\}.$$

Elements of $STS(G)$ are called (*terminal*) *structure trees* of G. When G is known, we sometimes speak simply about (Σ-)structure trees since the leaves are labeled by elements of Σ. Note that the rules of G determine the maximal number of immediate

successors of any node of a structure tree of G. Thus, of course, the alphabet Σ by itself does not determine the set of Σ-structure trees.

Grammars G_1 and G_2 are said to be *language equivalent* if $L(G_1)=L(G_2)$. It is well known that language equivalence is undecidable already for context-free grammars. Here we shall consider the following two more restricted notions of equivalence. Let G_1 and G_2 be ET0L grammars. The grammars G_1 and G_2 are

- *structurally equivalent* if $\mathrm{STS}(G_1)=\mathrm{STS}(G_2)$, and
- *syntax equivalent* if $\mathrm{TS}(G_1)$ and $\mathrm{TS}(G_2)$ are equal modulo a renaming of the nonterminals.

Note that syntax equivalence implies structural equivalence, and structurally equivalent grammars in turn are always language equivalent. Syntax equivalence is incomparable with the notion of *strong structural equivalence* [11] for ET0L grammars. Both syntax and structural equivalence are decidable for context-free and E0L grammars [8, 13, 15, 16, 19, 26]. (We have not formally defined these notions for sequential context-free grammars here, but the definitions are analogous to the parallel case.) Syntax equivalence and strong structural equivalence are decidable also for ET0L grammars but ET0L structural equivalence is undecidable [11, 24]. Here we will consider the structural equivalence problem for the deterministic ET0L grammars.

4. The main result

We show that EDT0L structural equivalence can be decided effectively. First we introduce some notations concerning the width of trees. Intuitively, a structure tree is said to have width M if it has at most M distinct subtrees at any level.

Definition 4.1. A set of Σ-structure trees $\{t_1,\ldots,t_m\}$ is said to have *subtree-width* M ($\in\mathbb{N}$) if, for all $0\leqslant k\leqslant \max_{1\leqslant j\leqslant m}\mathrm{hg}(t_j)$, the trees $t_1,\ldots,t_m$ have at most M distinct subtrees at level k, i.e., $\#\bigcup_{j=1}^{m}\mathrm{sub}_k(t_j)\leqslant M$.

In the above definition, note that $\mathrm{sub}_k(t_j)$ is a set of $(\Sigma\cup\{\sigma\})$-labeled trees, i.e., its elements do not consist of occurrences of subtrees in t_j. Note also that the subtree-width M does not need to be the minimal number of distinct subtrees at any given level, i.e., if $\{t_1,\ldots,t_m\}$ has width M then it has width M' for all $M'\geqslant M$.

We code the structure of an arbitrary derivation of an EDT0L grammar G_1 as a string. Then using the string encoding in its stack, a tree pushdown automaton can verify in one computation that *no* control-sequence of another grammar G_2 generates the same structure tree. The construction relies essentially on the fact that the failure of an EDT0L derivation with respect to a given control-sequence can be checked by following only one (nondeterministically chosen) path of the tree.

Lemma 4.1. *Let $G=(V,\Sigma,s_0,H)$ be an EDT0L grammar. Then there exists $M\in\mathbb{N}$ such that, for every control-sequence $\omega\in H^*$, the structure tree* $\mathbf{STR}(G(\omega))$ *has width M.*

Proof. Since the tables of H are homomorphisms, it follows that always when nodes $u_1, u_2 \in \mathrm{dom}(G(\omega))$ have the same length and are labeled by the same nonterminal, then $G(\omega)/u_1 = G(\omega)/u_2$. By choosing $M = |V \cup \Sigma| + 1$ we see that $\mathbf{STR}(G(\omega))$ has width M. □

Structure trees having constant subtree-width can be coded as strings where the ith symbol from the left codes the information which of the level i subtrees are direct descendants of which of the level $i-1$ nodes. The ith symbol also codes the order of occurrences of level i subtrees. The number of distinct level i subtrees is bounded by a constant and, furthermore, we need to consider only a constant number of level $i-1$ nodes that correspond to pairwise different subtrees.

Below we define the above described coding of structure trees having subtree-width M and prove a regularity property of such codings (in Lemma 4.3) for propagating EDT0L grammars only. The restriction to propagating grammars is done just to avoid unnecessarily complicated notations. Afterwards we explain how the result can be straightforwardly extended for grammars allowing erasing productions.

Definition 4.2. Let $G = (V, \Sigma, s_0, H)$ be a propagating EDT0L grammar and let

$$r = \max\{|h(a)| \mid h \in H, a \in V\}.$$

For each $M \in \mathbb{N}$ we define the set $\Omega(M)$ to consist of tuples

$$(m_1, m_2, \chi), \tag{4}$$

where $1 \leqslant m_1, m_2 \leqslant M$ and χ is a mapping of $\{1, \ldots, m_1\}$ into $\bigcup_{j=1}^{r} \{1, \ldots, m_2\}^j$ such that

$$\text{every element of } \{1, \ldots, m_2\} \text{ occurs in some tuple } \chi(j), \quad 1 \leqslant j \leqslant m_1. \tag{5}$$

The set of final symbols $\Omega_f(M)$ is defined to consist of tuples

$$(m_1, \Sigma, \chi), \tag{6}$$

where $1 \leqslant m_1 \leqslant M$ and χ is a mapping of $\{1, \ldots, m_1\}$ into $\bigcup_{j=1}^{r} \Sigma^j$. Note that here Σ^j is the set of ordered j-tuples of elements of Σ (and not the set of strings of length j).

A sequence W in $\Omega(M)^* \Omega_f(M)$,

$$W = (m_{1,1}, m_{1,2}, \chi_1)(m_{2,1}, m_{2,2}, \chi_2) \ldots (m_{n-1,1}, m_{n-1,2}, \chi_{n-1})(m_{n,1}, \Sigma, \chi_n) \tag{7}$$

is said to be *well-formed* if $m_{i+1,1} = m_{i,2}$ for all $i = 1, \ldots, n-1$.

Consider an s-tuple of structure trees $(t_1, \ldots, t_s)$, $s \geqslant 1$. Then $\sigma(t_1, \ldots, t_s)$ denotes the structure tree where the level one subtrees, from left to right, are $t_1, \ldots, t_s$. (This is just the standard algebraic notation for trees where we allow the symbol σ to have variable arity.)

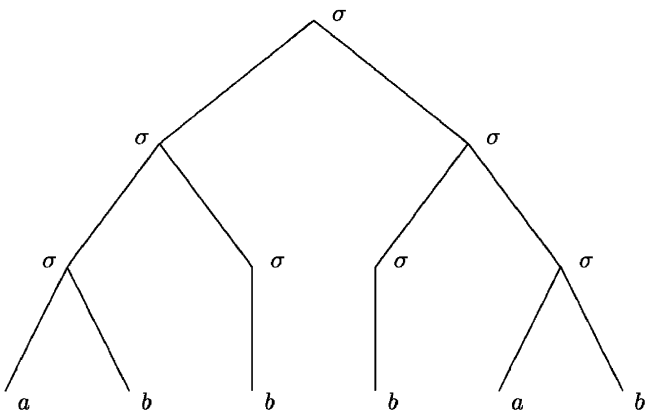

Fig. 1.

Corresponding to a well-formed sequence W as in (7) we define inductively an $m_{1,1}$-tuple of Σ-structure trees $\Xi(W)$. We say that the well-formed sequence W *represents* $\Xi(W)$. First a final symbol (6) represents the m_1-tuple of structure trees

$$\sigma(\chi(1)),\ldots,\sigma(\chi(m_1)).$$

For the inductive definition, denote by W' the suffix of W obtained by deleting the first symbol $(m_{1,1},m_{1,2},\chi_1)$ and assume that

$$\Xi(W')=(t_1,\ldots,t_{m_{2,1}}).$$

Note that $m_{1,2}=m_{2,1}$ since W is well-formed. Denote $\chi_1(j)=(a_{j,1},\ldots,a_{j,k_j})\in\{1,\ldots,m_{1,2}\}^{k_j}$, $1\leqslant k_j\leqslant r$, $j=1,\ldots m_{1,1}$. Then

$$\Xi(W)=(\sigma(t_{a_{1,1}},\ldots,t_{a_{1,k_1}}),\ldots,\sigma(t_{a_{m_{1,1},1}},\ldots,t_{a_{m_{1,1},k_{m_{1,1}}}})).$$

Example 4.1. Let $\Sigma=\{a,b\}$ and t is the tree given in Fig. 1. Choose

$$W=(1,2,\chi'')(2,2,\chi')(2,\Sigma,\chi)$$

where $\chi(1)=(a,b)$, $\chi(2)=(b)$, $\chi'(1)=(1,2)$, $\chi'(2)=(2,1)$ and $\chi''(1)=(1,2)$. Then $\Xi(W)=(t)$.

The following lemma can be proved using induction on the maximal height of the trees $t_1,\ldots,t_m$.

Lemma 4.2. *Let $M\in\mathbb{N}$ be fixed. Let $\{t_1,\ldots,t_m\}$ be a set of Σ-structure trees having subtree-width M such that* $\max_{1\leqslant i\leqslant m}\mathrm{hg}(t_i)=n$. *Then there exists a well-formed*

sequence $W \in \Omega(M)^{n-1}\Omega_f(M)$ *as in* (7) *such that*

$$\Xi(W) = (t_1, \ldots, t_m).$$

Furthermore, $m_{i,1}$ *is the cardinality of the set of level* $i-1$ *subtree representatives in* $\{t_1, \ldots, t_m\}$, $i = 1, \ldots, n$.

A word $W \in \Omega(M)^*\Omega_f(M)$ is said to be *simple* if the first symbol of W is of the form $(1, m, \chi)$, $m \leqslant M$. If W is simple and well-formed, then $\Xi(W)$ is a one-tuple (t), where t is a Σ-structure tree and, in practice, we identify $\Xi(W)$ with t.

As an immediate consequence of Lemma 4.2 we have

Corollary 4.1. *For every* Σ*-structure tree* t *having subtree-width* M *there exists a simple well-formed sequence* $W \in \Omega(M)^*\Omega_f(M)$ *such that* $\Xi(W) = t$.

The following lemma states that given a simple well-formed sequence $W \in \Omega(M)^*\Omega_f(M)$ and a control-sequence ω of an EDT0L grammar G, a finite automaton can determine whether or not $\Xi(W) = \mathbf{STR}(G(\omega))$.

Lemma 4.3. *Let* $G = (V, \Sigma, s_0, H)$ *be a propagating EDT0L grammar and* $M \in \mathbb{N}$. *Let the alphabets* $\Omega(M)$ *and* $\Omega_f(M)$ *be as in Definition* 4.2. *We denote by* L *the set of words* $\mu \in [(\Omega(M) \cup \Omega_f(M)) \times H]^+$ *such that*

(i) $\Pi_1(\mu) \in \Omega(M)^*\Omega_f(M)$ *is simple and well-formed,*

(ii) $\Xi(\Pi_1(\mu)) \neq \mathbf{STR}(G(\Pi_2(\mu)))$.

Here we denote the mappings $\Pi_i^{(\Omega(M) \cup \Omega_f(M)) \times H}$ *simply by* Π_i, $i = 1, 2$.

We claim that L *is a regular language.*

Proof. Condition (i) can be easily verified by a finite automaton. Hence it is sufficient to show that given $\mu \in [(\Omega(M) \cup \Omega_f(M)) \times H]^+$ satisfying (i), a finite automaton $\mathscr{A}$ can verify whether or not (ii) holds. (Note that if (i) does not hold, then $\Xi(\Pi_1(\mu))$ is not necessarily defined or may denote an ordered tuple of trees.)

The set of states of $\mathscr{A}$ is

$$\bigcup_{1 \leqslant i \leqslant M} [\wp(V) - \emptyset]^i \cup \{\mathbf{acc}, \mathbf{rej}\},$$

and the initial state is $\{s_0\}$. Assume that $\mathscr{A}$ is in a state $(U_1, \ldots, U_m)$, $\emptyset \neq U_i \subseteq V$, $1 \leqslant i \leqslant m \leqslant M$, and reading an input symbol

$$[(m_1, m_2, \chi), h].$$

If $m \neq m_1$, then $\mathscr{A}$ goes to the rejecting state **rej**. From our construction it follows that this is possible only when condition (i) does not hold. Assume then that $m = m_1$. If for some $x \in U_i$, $1 \leqslant i \leqslant m$,

$$\chi(i) \notin \{1, \ldots, m_2\}^{|h(x)|}, \tag{8}$$

then $\mathscr{A}$ goes to the accepting state **acc**. Also, $\mathscr{A}$ goes to the accepting state **acc** if for some $x \in U_i$, $1 \leqslant i \leqslant m$,

$$h(x) \notin V^+. \tag{9}$$

After this $\mathscr{A}$ reads the rest of the input verifying only that (i) holds.

The remaining possibility is that for all $x \in U_i$, $1 \leqslant i \leqslant m$, $\chi(i)$ is of the form $(s_1, \ldots, s_{|h(x)|})$, $s_j \in \{1, \ldots, m_2\}$, $j = 1, \ldots, |h(x)|$, and $h(x) \in V^+$. In this case after reading the symbol $[(m_1, m_2, \chi), h]$, $\mathscr{A}$ goes to the state $(Z_1, \ldots, Z_{m_2})$ where the sets $Z_j \subseteq V$, $j = 1, \ldots, m_2$, are constructed as follows. For each $i = 1, \ldots, m$ and each $x \in U_i$ do the following. If

$$h(x) = a_1 \ldots a_n \quad \text{and} \quad \chi(i) = (s_1, \ldots, s_n), \quad (s_j \in \{1, \ldots, m_2\}),$$

then add the element $a_j \in V$ to the set Z_{s_j}, $j = 1, \ldots, n$. Note that each of the sets $Z_1, \ldots, Z_{m_2}$ will be nonempty because χ satisfies the condition (5).

Intuitively, U_i consists of all elements of V that the derivation of G (following the morphisms $h' \in H$ read so far in the second components of the input) reaches at nodes corresponding to the ith subtree representative at this, say the kth, level of $\Xi(\Pi_1(\mu))$. Thus if condition (8) holds, the derivation reaches some level k node u of $\Xi(\Pi_1(\mu))$ with a symbol $a \in V$ such that the number of immediate successors of u is not equal to $|h(a)|$. If condition (9) holds, then $h(a)$ contains a terminal symbol. Thus a derivation using the table h cannot have the structure $\Xi(\Pi_1(\mu))$ and condition (ii) holds. The case where conditions (8) and (9) do not hold corresponds to the situation where the parallel derivation step determined by h at level k does not immediately violate the structure of the tree. Then the sets $Z_1, \ldots, Z_{m_2}$ are constructed to consist, respectively, of all nonterminals that will appear in the m_2 subtree representatives at the following level.

It remains to define the operation of $\mathscr{A}$ when it reaches a final symbol $[(m_1, \Sigma, \chi), h] \in [\Omega_f(M) \times H]$ in a state $(U_1, \ldots, U_m)$. Following the above idea this is done so that $\mathscr{A}$ accepts exactly then when the derivation step h produces for some leaf representative $b \in \Sigma$ a wrong terminal symbol (or a nonterminal).

The possibility $m \neq m_1$ corresponds again to a situation where (i) does not hold. We need to consider only the possibility $m = m_1$. If for all $i \in \{1, \ldots, m\}$ and for all $x \in U_i$, $\chi(i) = h(x)$, then the derivation step h produces correct terminal symbols at all leaves. This means that (ii) does not hold and $\mathscr{A}$ rejects. On the other hand, $\mathscr{A}$ enters the accepting state if for some $x \in U_i$, $\chi(i) \neq h(x)$. □

The above Lemma 4.3 was formulated and proved for propagating grammars only. However, exactly the same proof works also for general EDT0L grammars: the possibility of having erasing productions just adds at most one more subtree representative to each level of the structure tree. We can modify Definition 4.2 so that in the symbols (m_1, m_2, χ) (as in (4)) χ is a partial function where $\chi(i)$ is undefined if i represents a node having $\hat{\lambda}$ as the immediate successor (or in such cases $\chi(i)$ is defined to be a new symbol not belonging to $\{1, \ldots, m_2\}$.) The proof of Lemma 4.3 is then simply

modified by dividing the conditions (8) and (9) into cases depending on whether $\chi(i)$ is defined or not. Thus we can prove the following.

Lemma 4.4. *The statement of Lemma* 4.3 *holds without the assumption that G is propagating.*

Now we can show that a tree pushdown automaton can in a single computation verify whether all possible derivations of a given EDT0L grammar violate a given structure tree of constant width.

Lemma 4.5. *Let* $G_i = (V_i, \Sigma, s_{0,i}, H_i)$, $i = 1, 2$, *be EDT0L grammars. Let M be a constant guaranteed for* G_1 *by Lemma* 4.1. *Then we can effectively construct a tree pushdown automaton* **A** *such that* $T(\mathbf{A}) \neq \emptyset$ *if and only if there exists*

$$t \in STS(G_1) - STS(G_2) \tag{10}$$

such that t has width M.

Proof. Denote $\#H_2 = k$. The tree pushdown automaton **A** receives as inputs trees where each internal node has exactly k immediate successors. The internal nodes are labeled by a symbol a_0. The set of stack symbols is $(\Omega(M) \cup \Omega_f(M)) \times H_1$.

Assume that

$$t_1 \in \mathrm{STS}(G_1) - \mathrm{STS}(G_2), \tag{11}$$

where $\mathrm{hg}(t_1) = n + 1$, $n \geqslant 0$. Then the automaton **A** accepts the balanced k-ary input tree of height $n + 1$ as follows. At the beginning of the computation, **A** nondeterministically pushes into the stack a word $(\alpha_1, h_1) \ldots (\alpha_n, h_n)(\beta, h_{n+1})$, $\alpha_1, \ldots, \alpha_n \in \Omega(M)$, $\beta \in \Omega_f(M)$, $h_1, \ldots, h_{n+1} \in H_1$. (The top of the stack is to the left.)

Intuitively, the stack contents is guessed so that $W = \alpha_1 \ldots \alpha_n \beta$ $(\in \Omega(M)^* \Omega_f(M))$ is simple and well-formed and it satisfies the following property. If we denote $\omega = h_1 \ldots h_{n+1}$, then

$$\Xi(W) = \mathbf{STR}(G_1(\omega)) = t_1, \tag{12}$$

where t_1 is as in (11). By Corollary 4.1, there exists W satisfying (12).

When reading an input symbol, **A** always pops the topmost stack symbol and the remaining stack contents is forwarded to the k successor nodes. After the initial nondeterministic guesses **A** does not push any more symbols into the stack.

The states of **A** consist of two components that operate in parallel. The first component verifies that the condition (12) holds. As in the proof of Lemma 4.4 we see that this can be done using only a finite-state memory. The first component operates identically on all paths of the input tree, that is, it ignores the input symbols and treats the initial stack contents as input.

On the path to a leaf $i_1 i_2 \ldots i_{n+1}$ of the input, $1 \leqslant i_j \leqslant k$, $j = 1, \ldots, n+1$, the second component of **A** verifies that

$$\Xi(W) \neq \mathbf{STR}(G_2(i_1 i_2 \ldots i_{n+1})).$$

Again from the proof of Lemma 4.4 it follows that this is possible using the finite-state control of **A**. The second component of **A** ignores the second components $h_i \in H_1$ of the stack symbols.

Hence, on different paths of the input **A** verifies that $\Xi(W)$ is not the structure of a syntax tree $G_2(\omega')$ for any control-sequence ω' of length $n+1$, i.e., **A** verifies that $\Xi(W) \notin \mathrm{STS}(G_2)$. On the other hand, each branch of the computation has to consume the entire stack so an accepted input tree is necessarily balanced. Thus **A** accepts some input tree if and only if there exists a tree t such that (10) holds and t has subtree-width M. □

Note that in the proof of Lemma 4.5 it is essential that the guessed instance of $t \in \mathrm{STS}(G_1)$ in the pushdown stack has a string encoding, although the general tree pushdown automaton model of [9] allows, in fact, trees also in the stack. It can be shown that a tree pushdown automaton cannot nondeterministically push a balanced tree of arbitrary height into the stack [23], so one could not directly store an arbitrary $t \in \mathrm{STS}(G_1)$ in the tree stack at the beginning of the computation. Furthermore, simulating the derivations of G_2 given by different control-sequences directly on the tree t would have the following problem. The paths leading to "failure of the derivation of G_2 in t" may branch out earlier than the corresponding control-sequences branch out in the input tree. More specifically, the tree pushdown automaton **A** has to find, for each control-sequence ω of G_2, at least one path v_ω in t that leads to failure. The control-sequences correspond to different paths in the input and it is, in general, possible that two control-sequences ω_1 and ω_2 have a very long common prefix whereas the corresponding paths v_{ω_1} and v_{ω_2} in t branch out already at the root of t. Situations like this do not cause problems when **A** has a string encoding $\Xi(W)$ of t in the pushdown stack. Then **A** can simulate on all paths of the input all distinct subderivations of G_2 within the structure of t.

Combining Lemmas 4.1 and 4.5 and Proposition 2.1 we have proved the following result. Note that the constant M in Lemma 4.1 is independent of the control-sequence chosen.

Theorem 4.1. *For given EDT0L grammars G_1 and G_2 we can decide effectively whether or not*

$$STS(G_1) = STS(G_2).$$

Exactly as in the proof of Lemma 4.5, for given EDT0L grammars G_1 and G_2 the nonemptiness of $\mathrm{TS}(G_1) - \mathrm{TS}(G_2)$ can be reduced to deciding whether a tree pushdown automaton recognizes a nonempty tree language. Since the number of nonterminals of G_1 and G_2 is finite, we have a new proof for the decidability of the syntax equivalence. The result follows also from [24].

Theorem 4.2. *Syntax equivalence is decidable for EDT0L grammars.*

Table 1
Decidability of syntax and [strong] structural equivalence

	Syntax equiv.	Structural equiv.	Strong struct. equiv.
E0L	D	D	D
EDT0L	D	D	D
ET0L	D	U	D

5. Discussion and open problems

The decidability results for syntax equivalence, structural equivalence and strong structural equivalence of E(D)(T)0L grammars are summarized in Table 1. In the table D stands for "decidable" and U for "undecidable". The decidability of strong structural equivalence for ET0L grammars is proved in [11]. For E0L grammars, this notion coincides with structural equivalence. Language equivalence is naturally undecidable for all the cases. Note that the E0L and EDT0L language families are incomparable.

The proof of Theorem 4.1 gives only a multiple exponential time algorithm for EDT0L structural equivalence. We do not know what is the exact complexity of the problem. The deterministic exponential time hardness result obtained in [25] for E0L structural equivalence cannot be used, at least not directly, to prove a similar lower bound for the complexity of EDT0L structural equivalence. On the other hand, we cannot expect to obtain an efficient algorithm for the EDT0L case since it is known that already the structural equivalence problem for linear grammars is PSPACE-complete [10], and structural equivalence for linear grammars is easily logspace reducible to EDT0L structural equivalence. Note that when the sentential forms have only one nonterminal (or any constant number of occurrences of nonterminals), an EDT0L grammar can simulate a context-free derivation simply by having a different table for each rule.

Intuitively, the decidability proof of the previous section relies on the following two properties of EDT0L derivations:

(i) the current nonterminal and the remaining control-sequence determine uniquely a subderivation,

(ii) all control-sequences generate the structure tree one level at a time.

These properties enabled us to produce a string encoding W of a structure tree such that the failure of *all possible* control-sequences to produce this structure can be verified by a finite automaton that reads W and the control-sequence in parallel.

The necessity of both conditions (i) and (ii) can be illustrated by considering the *Indian parallel* (IP) grammars. An IP grammar is a context-free grammar with the derivation relation defined so that at each derivation step one rewrites all occurrences of one (nondeterministically chosen) nonterminal b in the given sentential form using the same rule with left side b. The other nonterminals are not rewritten. For the formal definition the reader may consult [2, 20, 27, 28]. It is well known that languages generated by IP grammars are strictly included in the EDT0L languages.

When the sequence of rules used in a derivation of an IP grammar is viewed as a control-sequence, the IP grammars clearly have the above property (i). However, different sequences of rules can generate distinct parts of a given structure tree in completely different order, and no analogy of condition (ii) seems to hold for IP grammars. Thus in spite of the fact that EDT0L grammars are strictly more powerful than IP grammars in terms of the family of generated languages, it does not appear possible to use the proof method of the previous section to decide the structural equivalence problem for IP grammars. We conjecture that IP structural equivalence is decidable. For the E0LIP grammars of [12] structural equivalence can be shown to be decidable exactly as in the proof of Theorem 4.1. (E0LIP grammars combine the Indian parallel and E0L rewriting mechanisms: at each derivation step all occurrences of every nonterminal are rewritten using the same rule.)

Russian parallel (RP) grammars [2, 14, 29] extend IP grammars by allowing also (sequential) context-free derivation steps. The decidability of the RP structural equivalence problem remains open.

References

[1] A.V. Aho, Indexed grammars: an extension of the context-free case, J. Assoc. Comput. Mach. 16 (1969) 383–406.
[2] J. Dassow, G. Păun, Regulated Rewriting in Formal Language Theory, EATCS Monographs in Theoretical Computer Science, Vol. 18, Springer, Berlin, 1989.
[3] J. Engelfriet, E.M. Schmidt, IO and OI, J. Comput. System Sci. 15 (1977) 328–353.
[4] J. Engelfriet, E.M. Schmidt, IO and OI, J. Comput. System Sci. 16 (1978) 67–99.
[5] M.J. Fischer, Grammars with macro-like productions, Proc. 9th Ann. IEEE Symp. on Switching and Automata Theory, 1968, pp. 131–142.
[6] F. Gécseg, M. Steinby, Tree automata, Akadémiai Kiadó, Budapest, 1984.
[7] F. Gécseg, M. Steinby, Tree languages, in: G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, Vol. III, Springer, Berlin, 1997, pp. 1–68.
[8] S. Ginsburg, M. Harrison, Bracketed context-free languages, J. Comput. System Sci. 1 (1967) 1–23.
[9] I. Guessarian, Pushdown tree automata, Math. Systems Theory 16 (1983) 237–263.
[10] H.B. Hunt III, D.J. Rosenkrantz, T.G. Szymanski, On the equivalence, containment, and covering problems for the regular and context-free languages, J. Comput. System Sci. 12 (1976) 222–268.
[11] G. Istrate, The strong equivalence of ET0L grammars, in: G. Rozenberg, A. Salomaa (Eds.), Developments in Language Theory, World Scientific, Singapore, 1994, pp. 81–89.
[12] H.C.M. Kleijn, G. Rozenberg, A study in parallel rewriting systems, Inform. and Control 44 (1980) 134–163.
[13] D.E. Knuth, A characterization of parenthesis languages, Inform. and Control 11 (1967) 269–289.
[14] M.K. Levitina, On some grammars with global productions, Akad. Nauk. SSSR NTI 2 (3) (1972) 32–36, (in Russian).
[15] R. McNaughton, Parenthesis grammars, J. Assoc. Comput. Mach. 14 (1967) 490–500.
[16] V. Niemi, A normal form for structurally equivalent E0L grammars, in: G. Rozenberg, A. Salomaa (Eds.), Lindenmayer Systems: Impacts on Theoretical Computer Science, Computer Graphics, and Developmental Biology, Springer, Berlin, 1992, pp. 133–148.
[17] Th. Ottmann, D. Wood, Defining families of trees with E0L grammars, Discrete Applied Math. 32 (1991) 195–209.
[18] Th. Ottmann, D. Wood, Simplifications of E0L grammars, in: G. Rozenberg, A. Salomaa (Eds.), Lindenmayer Systems: Impacts on Theoretical Computer Science, Computer Graphics, and Developmental Biology, Springer, Berlin, 1992, pp. 149–166.

[19] M. Paull, S. Unger, Structural equivalence of context-free grammars, J. Comput. System Sci. 2 (1968) 427–463.
[20] B. Reichel, Some classifications of Indian parallel languages, J. Inform. Process. Cybernet. EIK 26 (1990) 85–99.
[21] W.C. Rounds, Mappings and grammars on trees, Math. Systems Theory 4 (1970) 257–287.
[22] G. Rozenberg, A. Salomaa, The Mathematical Theory of L Systems, Academic Press, New York, 1980.
[23] K. Salomaa, Deterministic tree pushdown automata and monadic tree rewriting systems, J. Comput. System Sci. 37 (1988) 367–394.
[24] K. Salomaa, D. Wood, S. Yu, Structural equivalence and ET0L grammars, Proc. 9th Conf. on Fundamentals of Computation Theory, FCT'93, Lecture Notes in Computer Science 710, Springer, Berlin, 1993, pp. 430–439. The full version appeared in: Theoret. Comput. Sci. 164 (1996) 123–140.
[25] K. Salomaa, D. Wood, S. Yu, Complexity of E0L structural equivalence, RAIRO Theoret. Inform. 29 (1995) 471–485.
[26] K. Salomaa, S. Yu, Decidability of structural equivalence of E0L grammars, Theoret. Comput. Sci. 82 (1991) 131–139.
[27] R. Siromoney, K. Krithivasan, Parallel context-free languages, Inform. and Control 24 (1974) 155–162.
[28] S. Skyum, Parallel context-free languages, Inform. and Control 26 (1974) 280–285.
[29] S. Skyum, Decomposition theorems for various kinds of languages parallel in nature, SIAM J. Comput. 5 (1976) 284–296.
[30] D. Wood, Theory of Computation, Wiley, New York, 1987.